The program-size complexity of self-assembled paths

Pierre-Etienne Meunier, Damien Regnault, Damien Woods

The Program-Size Complexity of Self-Assembled Paths*

Pierre-Étienne Meunier
Hamilton Institute,
Department of Computer Science,
Maynooth University
Maynooth, Ireland
pmeunier@mailbox.org

Damien Regnault
IBISC, Université Évry, Université
Paris-Saclay
91025, Evry, France
damien.regnault@univ-evry.fr

Damien Woods
Hamilton Institute,
Department of Computer Science,
Maynooth University
Maynooth, Ireland
damien.woods@mu.ie

ABSTRACT

We prove a Pumping Lemma for the *noncooperative* abstract Tile Assembly Model, a model central to the theory of algorithmic self-assembly since the beginning of the field. This theory suggests, and our result proves, that small differences in the nature of adhesive bindings between abstract square molecules gives rise to vastly different expressive capabilities.

In the *cooperative* abstract Tile Assembly Model, square tiles attach to each other using multi-sided cooperation of one, two or more sides. This precise control of tile binding is directly exploited for algorithmic tasks including growth of specified shapes using very few tile types, as well as simulation of Turing machines and even self-simulation of self-assembly systems. But are cooperative bindings required for these computational tasks? The definitionally simpler *noncooperative (or Temperature 1)* model has poor control over local binding events: tiles stick if they bind on at least one side. This has led to the conjecture that it is impossible for it to exhibit precisely controlled growth of computationally-defined shapes.

Here, we prove such an impossibility result. We show that any planar noncooperative system that attempts to grow large, tile-efficient assemblies in an algorithmic way must also grow infinite non-algorithmic (pumped) structures with a simple closed-form description, or else suffer blocking of intended algorithmic structures. Our result holds for both directed and nondirected systems, and gives an explicit upper bound of $(8|T|)^{4|T|+1}(5|\sigma|+6)$, where $|T|$ is the size of the tileset and $|\sigma|$ is the size of the seed assembly, beyond which any path of tiles is pumpable or blockable.

CCS CONCEPTS

• **Mathematics of computing**; • **Theory of computation** → **Models of computation**; **Computational geometry**;

KEYWORDS

Self-assembly, tilings, pumping lemma, DNA computing

ACM Reference Format:
Pierre-Étienne Meunier, Damien Regnault, and Damien Woods. 2020. The Program-Size Complexity of Self-Assembled Paths. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20), June 22–26, 2020, Chicago, IL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3357713.3384263

1 INTRODUCTION

Disclaimer: The version of this work you are reading is merely a brief Extended Abstract, the full paper [45] can be found on the arxiv (arXiv:2002.04012v1 [cs.CC]) and contains proofs, figures and additional intuitive explanations.

The main challenge of molecular programming is to understand, build and control matter at the molecular level. The dynamics of molecules can embed algorithms [42] and the theory of algorithmic self-assembly [13, 34, 43] is one formal way to think about the computational capabilities of autonomic self-assembling molecular systems. That theory, and more broadly the theory of models of computation, guides advances in experimental work to this day: the self-assembling binary counter of Winfree and Rothemund [38] was later implemented using tiles made of DNA [17], as were bit-copying systems [2, 3, 39] and discrete self-similar fractals [20, 36]. More recently, twenty-one self-assembly algorithms were implemented using DNA single-stranded tiles [44], including a tile-based implementation [7] of von Neuman's fair-bit-from-an-unfair-coin and a 3-bit instance of the computationally universal cellular automata Rule 110 [9, 32]. Besides guiding experiment, the theory itself has undergone significant developments, with the long-term vision of understanding the kinds of structure-building capabilities and computational mechanisms that are implementable and permitted by molecular processes.

Perhaps the most studied model of algorithmic self-assembly is the abstract Tile Assembly Model (aTAM), introduced by Winfree [42] as a computational model of DNA tile-based self-assembly. The model is an algorithmic version of Wang tilings [41], can be thought of as an asynchronous cellular automaton [5] and has features seen in other distributed computing models. In each instance of the model, we have a finite set of unit square tile *types*, with colours on their four sides, and an infinite supply of each type. Starting from a small connected arrangement of tiles on the $\mathbb{Z}^2$ plane, called the *seed assembly*, we attach tiles to that assembly asynchronously and nondeterministically based on a local rule depending only on the colour of the sides of the newly placed tile and

*Supported by European Research Council (ERC) award number 772766 and Science foundation Ireland (SFI) grant 18/ERCS/5746 (this manuscript reflects only the authors' view and the ERC is not responsible for any use that may be made of the information it contains). Some of this work was supported by, and carried out at, Inria, Paris, France. This version is merely an Extended Abstract, the full paper can be found on the arxiv at arXiv:2002.04012v1 [cs.CC] and contains proofs, figures and additional intuitive explanations.

STOC '20, June 22–26, 2020, Chicago, IL, USA

ACM ISBN 978-1-4503-6979-4/20/06…$15.00
https://doi.org/10.1145/3357713.3384263

of the sides of the assembly that are adjacent to the attachment position.

This model can simulate Turing machines [42], self-assemble squares with few tile types [37, 38], assemble any finite spatially-scaled shape using a small, Kolmogorov-efficient, tile set [40], and there is an intrinsically universal tile set capable of simulating the behaviour (produced shapes and growth dynamics) of any other tile set, up to spatial rescaling [14].

However, these results have all been proven using the so-called *cooperative* tile assembly model. In the cooperative, or temperature 2, model there are two kinds of bonds: strong and weak. A tile can attach to an assembly by one side if that side forms a strong ("strength 2") bond with the assembly, or it can attach if two of its sides each match with a weak ("strength 1") bond. Intuitively, the cooperative model exploits weak bonds to create a form of synchronisation. The attachment of a tile by two weak glues sticking to two neighbour tiles can only occur *after* both neighbour tiles are present, allowing the system to *wait* rather than (say) proceeding with potentially inaccurate or incomplete information.

But what happens if we allow only one kind of bond? We get a simple model called the *noncooperative*, or *temperature 1*, model. In the noncooperative model, tiles may attach to the assembly whenever at least one of their sides' colour matches the colour of a side of the assembly adjacent to the position where they attach. Intuitively, they need not wait for more bonds to appear adjacent to their attachment position. In this model it seems non-obvious how to implement synchronisation; we have no *obvious* programmable feature that enables one growth process to wait until another is complete. Our attempts to build such things typically lead to rampant uncontrolled growth.

The question of whether the noncooperative (or "non-waiting") model has *any* non-trivial computational abilities has been open since the beginning of the field [38]. Perhaps a reason for this is that actually proving that one can not synchronise growth is tricky; maybe noncooperative self-assembly can somehow simulate synchronisation using some complicated form of in-plane geometric blocking? Restrictions of the model have been shown to be extremely weak [26, 38], generalisations shown to be extremely powerful [4, 6, 10–12, 15, 18, 19, 23–25, 33, 35, 37], and, to further deepen the mystery, the model has been shown capable of some (albeit limited) efficient tile reuse [27, 30].

1.1 Main Result

Our main result is stated in Theorem 1.1, although a number of notions have yet to be formally defined (see Section 2 for definitions). Intuitively if a noncooperative tile assembly system produces a large assembly, it is capable of also producing any path of tiles in that assembly. Our statement says that if the tile assembly system can produce a long enough path P, then it must also produce assemblies where either an infinite ultimately periodic path appears (P is "pumpable"), or else the path cannot appear in all terminal assemblies (because some other tiles can be placed to block the growth of P), in which case we say that P is *fragile*. Let T be a set of tile types, and let $|\sigma|$ denote the number of tiles in the (seed) assembly σ.

Theorem 1.1. *Let $\mathcal{T} = (T, \sigma, 1)$ be any tile assembly system in the abstract Tile Assembly Model (aTAM), and let P be a path producible by $\mathcal{T}$. If P has vertical height or horizontal width at least $(8|T|)^{4|T|+1}(5|\sigma| + 6)$, then P is pumpable or fragile.*

Our result rules out noncooperative implementation of the kind of Turing machine simulations, and other kinds of computations, that have appeared in the literature to date and execute precisely controlled growth patterns [4, 6, 10–12, 15, 18, 19, 23–26, 33, 35, 37]. We do so by showing that any attempt to noncooperatively carry out long computations such as these, which provably require large assemblies and thus long paths, will result in unbounded pumped growth or the blocking of such paths.

The essence of algorithmic self-assembly is tile reuse: growing structures that are larger than the number of available tile types [13, 34, 38, 43]. Meunier and Regnault [30] show that some noncooperative systems are capable of tile reuse in the following sense: there is a tile assembly system with multiple terminal assemblies, all of finite size, such that each of them contains the same long path P, where P is of width $O(|T| \log |T|)$ (i.e. larger than $|T|$). In that construction, P is neither pumpable (all assemblies are finite) nor fragile (P appears in all terminal assemblies, hence no assembly or path can block it). Their result should be contrasted with ours since here we show that any attempt to generalise such a construction beyond our exponential-in-$|T|$ bound will fail – thus we give a limitation of the amount of tile reuse possible in such constructions. Analogous tile reuse limitations do not appear in cooperative systems [38], due to their ability to run arbitrary algorithms.

Our theorem statement is quite similar to the pumpability conjecture of Doty, Patitz and Summers [16] (Conjecture 6.1). In that work [16] under the assumption that the conjecture is true, they achieve a complete characterisation of producible assemblies. Our result is slightly different from that conjecture, being stronger in two ways, and weaker in one:

- First, their conjecture was stated for *directed* systems (that produce a single terminal assembly), but here we prove the result for both directed and undirected systems (systems that produce many terminal assemblies).[1] In the directed case, our result shows that any attempt to simulate computations by growing longer and longer paths is doomed to also produce a terminal assembly littered with more and more infinite pumped paths.
- Second, we give an explicit bound (exponential in $|T|$) on the length a path can reach before it is pumpable or fragile.
- Our result is weaker in one way: indeed, while our result only applies to paths grown all the way from the seed, the conjecture is that arbitrary paths are pumpable, regardless of their position relative to the seed.
  However, we conjecture that our result is sufficient to achieve the same characterisation of producible assemblies, using the same techniques and arguments as [16].

Our result can be applied to other models. After the aTAM, another well-studied model in the theory of algorithmic self-assembly is the hierarchical, or two-handed, model (2HAM) [6, 8, 12, 15]. There is no seed assembly in the 2HAM: in the noncooperative

[1] Directed systems do not have fragile paths, hence for directed systems the conclusion of the theorem statement systems simply reads "... then P is pumpable".

(temperature 1) 2HAM tiles stick together if glues match on one tile side, forming a collection of assemblies, and those assemblies can in turn stick to each other if they can be translated to touch without overlapping and with adjacent matching glues between them. As an almost direct corollary of Theorem 1.1 we get:

Corollary 1.2. *Let $\mathcal{H} = (T, 1)$ be any tile assembly system in the Two-Handed Assembly Model (2HAM), and let P be a path producible by $\mathcal{H}$. If P has vertical height or horizontal width at least $(88|T|)^{4|T|+1}$, then P is pumpable or fragile.*

Intuitively, the corollary comes from the fact that aTAM-like growth is permitted in the 2HAM. In fact, if we fix a tile set T and a temperature of 1, the set of producible assemblies in a 2HAM system over T is a superset (sometimes a strict superset [6]) of those in the aTAM over T. A brief proof sketch is given an Appendix to the full version of this paper [45].

1.2 Relationship With Other Prior Work

Perhaps due to the difficulty of analysing the standard noncooperative model (2D, square tiles, tiles attach one at a time), researchers have looked at different variants of that model.

The first restriction studied was where we permit only terminal assemblies that are "fully connected" meaning all tiles are fully bound to all of their neighbour tiles): Rothemund and Winfree [38] showed that for each large enough $n \in \mathbb{N}$ there does not exist a noncooperative system that builds a fully-connected $n \times n$ square in a tile-efficient way (using $< n^2$ tile types). Since embedding algorithms in tiles are essentially our best (and perhaps only) way to exhibit efficient tile reuse, our result shuts the door on a wide class of algorithmic approaches. Other restrictions proven weak are where we disallow adjacent mismatching colours [26], or even force any pair of adjacent tiles to bind to each other [38].

Another, quite productive, approach has been to study generalisations of the noncooperative model (e.g. 3D, non-square tiles, multi-tile assembly steps, more complicated 'active' tiles, etc.): it turns out that these generalisations and others are powerful enough to simulate Turing machines [4, 6, 10–12, 15, 18, 19, 21–26, 33, 35, 37]. Each such result points to a specific feature or set of features in a generalised model that provably needs to be exploited in order to avoid our negative result.

Cook, Fu, and Schweller [10] have shown that for any Turing machine there is an undirected tile assembly system, whose seed encodes an input, and where the largest producible terminal assembly (which is possibly infinite) simulates the Turing machine computation on that input. However, in that construction, "blocking errors" can occur where growth is prematurely blocked and is stopped before the simulation, or computation, is completed (hence their result is stated in a probabilistic setting). Indeed their tile assembly systems that simulate Turing machines will produce many such erroneous assemblies. Our result shows that this kind of blocking is unavoidable.

Assemblies that cannot be "blocked" have the opposite issue, where it seems there is always a part of the assembly that can be repeated forever, which led Doty, Patitz and Summers [16] to their pumpability conjecture. They go on to show that, assuming the pumpability conjecture holds, projections to the vertical/horizontal axes of assemblies produced by directed noncooperative systems have a straightforward closed-form description (as the union of semi-linear sets).

In the direction of negative results on Temperature 1, reference [31] showed that the noncooperative planar aTAM is not capable of simulating – in the sense used in intrinsic universality [43] – other noncooperative aTAM systems. In other words intrinsic universality is not possible for the planar noncooperative model. The Temperature 2 (or, cooperative) model is capable of such simulations [14], hence the main result of [31] shows a difference in the self-simulation capabilities of the two models. Prior to that work, another result showed that Temperature 1 cannot simulate Temperature 2 intrinsically [28], and hence the former model is strictly weaker than the later in this setting (where we ignore spatial scaling). However, to obtain both of those results, the use of simulation and intrinsic universality permitted a technique where we choose a particular class of shapes we want to simulate, and restrict the analysis of produced assemblies to (scaled versions of) these shapes. The proofs [28, 31] then involved forcing certain paths to grow outside of these pre-determined shapes. Here, we make use of a number of tools from [31]. However, the setting here is significantly more challenging as we have no geometric hypotheses whatsoever on producible assemblies and therefore can not directly leverage [31], although we do make use of the tools of visibility and the notion of right/left priority exploited in that prior work. As already noted, *positive* constructions have been found that some limited form of efficient reuse of tile types is possible in the standard 2D noncooperative model [27, 30]. This paper also builds on extensive previous work by two of the authors [29].

1.3 New Tools and Future Work

We develop a collection of new tools to reason about paths in $\mathbb{Z}^2$. In order to carry out computation in tile-assembly, a key idea is to reuse tile types (analogous to how a Boolean circuit reuses gates of a given type, or how a computer program reuses instructions via loops). Our main technical lemma, which we term the "shield" lemma, shows that if a path P has a certain form that reaches so far to the east that it reuses some tile types, then we can use P to construct a curve in $\mathbb{R}^2$ that is an almost-vertical cut c of the plane. We use this cut of the plane to show one of two things must happen: either that iterations of a pumping of P are separated from one another, and hence that the pumping is simple, which means that P is pumpable, or else that a path can be grown that blocks the growth of P.

This cut, and our subsequent argument can be thought of as a kind of "Window Movie Lemma" [28], or pumping tool, but targeted specifically at noncooperative self-assembly.

The shield lemma can only be applied when P is of a certain form. Our second tool is a combinatorial argument on the height and width of a path. We begin by applying some straight-froward transformations to put any wide enough or tall enough path (i.e., that meets the hypothesis of Theorem 1.1) into a form where its last tile is also its eastern-most tile. Then, for any such path P, our combinatorial argument shows that we can always find a cut that satisfies the hypothesis of the shield lemma. We hope that the

techniques developed in this paper can be applied to other self-assembly models – as an example we apply them to the 2HAM (Corollary 1.2).

Although we answer one of the main unresolved questions on noncooperative self-assembly, our result does not close all questions on the model. First, there is still a large gap between the best known lower bounds on the size of assemblies, which was shown in [30] is $O(|T| \log |T|)$, and the bound $(8|T|)^{4|T|+1}(5|\sigma|+6)$ we prove here. It remains as future work to reduce that gap. We also conjecture that our result can be used to characterise the assemblies producible by directed systems, using the same argument as [16]. A number of decidability questions are also still open such as: can we decide whether a planar nondirected tile assembly system is directed (i.e. produces exactly one terminal assembly)? An important part of self-assembly is related to building shapes (as opposed to decidability questions). In this direction, can we build $n \times n$ squares any more efficiently than the best known result of $2n - 1$ tile types for noncooperative systems? For this common benchmark of shape-building, cooperative systems achieve a tile-set size as low as $\Theta(\log n / \log \log n)$ [1, 22, 38].

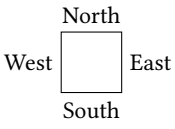
2 DEFINITIONS AND PRELIMINARIES

As usual, let $\mathbb{R}$ be the set of real numbers, let $\mathbb{Z}$ be the integers, and let $\mathbb{N}$ be the natural numbers including 0. The domain of a function f is denoted $\mathrm{dom}(f)$, and its range (or image) is denoted $f(\mathrm{dom}(f))$.

2.1 Abstract Tile Assembly Model

The abstract tile assembly model was introduced by Winfree [42]. In this paper we study a restriction of the abstract tile assembly model called the temperature 1 abstract tile assembly model, or noncooperative abstract tile assembly model. For definitions of the full model, as well as intuitive explanations, see for example [37, 38].

A *tile type* is a unit square with four sides, each consisting of a glue *type* and a nonnegative integer *strength*. Let T be a a finite set of tile types. In any set of tile types used in this paper, we assume the existence of a well-defined total ordering which we call the *canonical ordering* of the tile set. The sides of a tile type are respectively called north, east, south, and west, as shown in the following picture:



An *assembly* is a partial function $\alpha : \mathbb{Z}^2 \dashrightarrow T$ where T is a set of tile types and the domain of α (denoted $\mathrm{dom}(\alpha)$) is connected.[2] We let $\mathcal{A}^T$ denote the set of all assemblies over the set of tile types T. In this paper, two tile types in an assembly are said to *bind* (or *interact*, or are *stably attached*), if the glue types on their abutting sides are equal, and have strength ≥ 1. An assembly α induces an undirected weighted *binding graph* $G_\alpha = (V, E)$, where $V = \mathrm{dom}(\alpha)$, and there is an edge $\{a, b\} \in E$ if and only if the tiles at positions a and b interact, and this edge is weighted by the glue strength of that interaction. The assembly is said to be τ-stable if every cut of G_α has weight at least τ.

A *tile assembly system* is a triple $\mathcal{T} = (T, \sigma, \tau)$, where T is a finite set of tile types, σ is a τ-stable assembly called the *seed*, and $\tau \in \mathbb{N}$ is the *temperature*. Throughout this paper, $\tau = 1$.

Given two τ-stable assemblies α and β, we say that α is a *subassembly* of β, and write $\alpha \sqsubseteq \beta$, if $\mathrm{dom}(\alpha) \subseteq \mathrm{dom}(\beta)$ and for all $p \in \mathrm{dom}(\alpha)$, $\alpha(p) = \beta(p)$. We also write $\alpha \to_1^{\mathcal{T}} \beta$ if we can obtain β from α by the binding of a single tile type, that is: $\alpha \sqsubseteq \beta$, $|\mathrm{dom}(\beta) \setminus \mathrm{dom}(\alpha)| = 1$ and the tile type at the position $\mathrm{dom}(\beta) \setminus \mathrm{dom}(\alpha)$ stably binds to α at that position. We say that γ is *producible* from α, and write $\alpha \to^{\mathcal{T}} \gamma$ if there is a (possibly empty) sequence $\alpha_1, \alpha_2, \ldots, \alpha_n$ where $n \in \mathbb{N} \cup \{\infty\}$, $\alpha = \alpha_1$ and $\alpha_n = \gamma$, such that $\alpha_1 \to_1^{\mathcal{T}} \alpha_2 \to_1^{\mathcal{T}} \ldots \to_1^{\mathcal{T}} \alpha_n$. A sequence of $n \in \mathbb{Z}^+ \cup \{\infty\}$ assemblies $\alpha_0, \alpha_1, \ldots$ over $\mathcal{A}^T$ is a *$\mathcal{T}$-assembly sequence* if, for all $1 \leq i < n$, $\alpha_{i-1} \to_1^{\mathcal{T}} \alpha_i$.

The set of *productions*, or *producible assemblies*, of a tile assembly system $\mathcal{T} = (T, \sigma, \tau)$ is the set of all assemblies producible from the seed assembly σ and is written $\mathcal{A}[\mathcal{T}]$. An assembly α is called *terminal* if there is no β such that $\alpha \to_1^{\mathcal{T}} \beta$. The set of all terminal assemblies of $\mathcal{T}$ is denoted $\mathcal{A}_\square[\mathcal{T}]$.

2.2 Paths and Noncooperative Self-Assembly

Let T be a set of tile types. A *tile* is a pair $((x, y), t)$ where $(x, y) \in \mathbb{Z}^2$ is a position and $t \in T$ is a tile type. Intuitively, a path is a finite or one-way-infinite simple (non-self-intersecting) sequence of tiles placed on points of $\mathbb{Z}^2$ so that each tile in the sequence interacts with the previous one, or more precisely:

Definition 2.1 (Path). A *path* is a (finite or infinite) sequence $P = P_0 P_1 P_2 \ldots$ of tiles $P_i = ((x_i, y_i), t_i) \in \mathbb{Z}^2 \times T$, such that:

- for all P_j and P_{j+1} defined on P, their positions (x_j, y_j) and (x_{j+1}, y_{j+1}) are adjacent nodes in the grid graph of $\mathbb{Z}^2$, moreover t_j and t_{j+1} interact (have matching glues on their abutting sides), and
- for all P_j, P_k such that $j \neq k$ it is the case that $(x_j, y_j) \neq (x_k, y_k)$.

By definition, paths are simple (or self-avoiding), and this fact will be repeatedly used throughout the paper. For a tile P_i on some path P, its x-coordinate is denoted x_{P_i} and its y-coordinate is denoted y_{P_i}. The *concatenation* of two paths P and Q is the concatenation PQ of these two paths as sequences, and PQ is a path if and only if (1) the last tile of P interacts with the first tile of Q and (2) P and Q do not intersect each other.

For a path $P = P_0 \ldots P_i P_{i+1} \ldots P_j \ldots$, we define the notation $P_{i,i+1,\ldots,j} = P_i P_{i+1} \ldots P_j$, i.e. "the subpath of P between indices i and j, inclusive". Whenever P is finite, i.e. $P = P_0 P_1 P_2 \ldots P_{n-1}$ for some $n \in \mathbb{N}$, n is termed the *length* of P and denoted by $|P|$. In the special case of a subpath where $i = 0$, we say that $P_{0,1,\ldots,j}$ is a prefix of P and that P is *an extension* of $P_{0,1,\ldots,j}$. The prefix or extension are said to be *strict* if $j < |P| - 1$. Else, when $j = |P| - 1$, we say that $P_{i,\ldots,|P|-1}$ is a suffix of P, and is a *strict suffix* of P if $i > 0$.

For any path $P = P_0 P_1 P_2, \ldots$ and integer $i \geq 0$, we let $\mathrm{pos}(P_i) \in \mathbb{Z}^2$, or $(\mathrm{x}_{P_i}, \mathrm{y}_{P_i}) \in \mathbb{Z}^2$, for the position of P_i and $\mathrm{type}(P_i)$ for the tile type of P_i. Hence if $P_i = ((x_i, y_i), t_i)$ then $\mathrm{pos}(P_i) = (\mathrm{x}_{P_i}, \mathrm{y}_{P_i}) =$

[2] Intuitively, an assembly is a positioning of unit-sized tiles, each from some set of tile types T, so that their centers are placed on (some of) the elements of the discrete plane $\mathbb{Z}^2$ and such that those elements of $\mathbb{Z}^2$ form a connected set of points.

(x_i, y_i) and $\text{type}(P_i) = t_i$. A "*position of P*" is an element of $\mathbb{Z}^2$ that appears in P (and therefore appears exactly once), and an *index i* of a path P of length $n \in \mathbb{N}$ is a natural number $i \in \{0, 1, \ldots, n-1\}$. For a path $P = P_0P_1P_2\ldots$ we write $\text{pos}(P)$ to mean "the sequence of positions of tiles along P", which we define to be $\text{pos}(P) = \text{pos}(P_0)\text{pos}(P_1)\text{pos}(P_2)\ldots$.

Although a path is not an assembly, we know that each adjacent pair of tiles in the path sequence interact implying that the set of path positions forms a connected set in $\mathbb{Z}^2$ and hence every path uniquely represents an assembly containing exactly the tiles of the path, more formally: for a path $P = P_0P_1P_2\ldots$ we define the set of tiles $\text{asm}(P) = \{P_0, P_1, P_2, \ldots\}$ which we observe is an assembly[3] and we call $\text{asm}(P)$ a *path assembly*. Given a tile assembly system $\mathcal{T} = (T, \sigma, 1)$ the path P is a *producible path of* $\mathcal{T}$ if $\text{asm}(P)$ does not intersect[4] the seed σ and the assembly $(\text{asm}(P) \cup \sigma)$ is producible by $\mathcal{T}$, i.e. $(\text{asm}(P) \cup \sigma) \in \mathcal{A}[\mathcal{T}]$, and P_0 interacts with a tile of σ. As a convenient abuse of notation we sometimes write $\sigma \cup P$ as a shorthand for $\sigma \cup \text{asm}(P)$. Given a tile assembly system $\mathcal{T} = (T, \sigma, 1)$ we define the set of producible paths of $\mathcal{T}$ to be[5] $\text{P}[\mathcal{T}] = \{P \mid P$ is a path that does not intersect σ and $(\text{asm}(P) \cup \sigma) \in \mathcal{A}[\mathcal{T}]\}$.

So far, we have defined paths of tiles (Definition 2.1). In our proofs, we will also reason about (untiled) *binding paths* in the binding graph of an assembly.

Definition 2.2 (Binding path). Let $G = (V, E)$ be a binding graph. A *binding path* q in G is a sequence $q_{0,1,\ldots,|q|-1}$ of vertices from V such that for all $i \in \{0, 1, \ldots, |q| - 2\}$, $\{q_i, q_{i+1}\} \in E$ (q is connected) and no vertex appears twice in q (q is simple).

Observation 2.3. *Let $\mathcal{T} = (T, \sigma, 1)$ be a tile assembly system and let $\alpha \in \mathcal{A}[\mathcal{T}]$. For any tile $((x, y), t) \in \alpha$ either $((x, y), t)$ is a tile of σ or else there is a producible path $P \in \text{P}[\mathcal{T}]$ that for some $j \in \mathbb{N}$ contains $P_j = ((x, y), t)$.*

For $A, B \in \mathbb{Z}^2$, we define $\overrightarrow{AB} = B - A$ to be the vector from A to B, and for two *tiles* $P_i = ((x_i, y_i), t_i)$ and $P_j = ((x_j, y_j), t_j)$ we define $\overrightarrow{P_iP_j} = \text{pos}(P_j) - \text{pos}(P_i)$ to mean the vector from $\text{pos}(P_i) = (x_i, y_i)$ to $\text{pos}(P_j) = (x_j, y_j)$. The translation of a path P by a vector $\vec{v} \in \mathbb{Z}^2$, written $P + \vec{v}$, is the path Q such that $|P| = |Q|$ and for all indices $i \in \{0, 1, \ldots, |P| - 1\}$, $\text{pos}(Q_i) = \text{pos}(P_i) + \vec{v}$ and $\text{type}(Q_i) = \text{type}(P_i)$. We always use parentheses for scoping of translations when necessary, i.e. $P(Q + \vec{v})$ is the sequence containing the path P followed by the translation of the entire path Q by vector $\vec{v}$. The translation of an assembly α by a vector $\vec{v}$, written $\alpha + \vec{v}$, is the assembly β defined for all $(x, y) \in (\text{dom}(\alpha) + \vec{v})$ as $\beta(x, y) = \alpha((x, y) - \vec{v})$.

Let P be a path, let $i \in \{1, \ldots, |P|-2\}$ and $A \neq P_{i+1}$ be a tile such that $P_{0,1,\ldots,i}A$ is a path. Let also ρ be the clockwise rotation matrix defined as $\rho = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$, and let $\Delta = (\rho\overrightarrow{P_iP_{i-1}}, \rho^2\overrightarrow{P_iP_{i-1}}, \rho^3\overrightarrow{P_iP_{i-1}})$ (intuitively, Δ is the vector of possible steps after P_i, ordered clockwise). We say that $P_{0,1,\ldots,i}A$ *turns right* (respectively *turns left*) from $P_{0,1,\ldots,i+1}$ if $\overrightarrow{P_iA}$ appears after (respectively before) $\overrightarrow{P_iP_{i+1}}$ in Δ.

Definition 2.4 (The right priority path of a set of paths or binding paths). Let P and Q be two paths, where $P \neq Q$ and moreover neither is a prefix of the other, and with $\text{pos}(P_0) = \text{pos}(Q_0)$ and $\text{pos}(P_1) = \text{pos}(Q_1)$. Let i be the smallest index such that $i \geq 0$ and $P_i \neq Q_i$. We say that P is the *right priority path* of P and Q if either (a) $P_{0,1,\ldots,i}$ is a right turn from Q or (b) $\text{pos}(P_i) = \text{pos}(Q_i)$ and the type of P_i is smaller than the type of Q_i in the canonical ordering of tile types.

Similarly, let p and q be two binding paths, where $p \neq q$ and moreover neither is a prefix of the other, and with $p_0 = q_0$ and $p_1 = q_1$. Let i be the smallest index such that $i \geq 0$ and $p_i \neq q_i$. We say that p is the *right priority path* of p and q if $p_{0,1,\ldots,i}$ is a right turn from q.

For any finite set S of paths, or of binding paths, we extend this definition as follows: let $p_0 \in \mathbb{Z}^2, p_1 \in \mathbb{Z}^2$ be two adjacent positions. If for all $s \in S$, we have $s_0 = p_0$ and $s_1 = p_1$, we call the *right-priority path of S* the path that is the right-priority path of all other paths in S.

For all $i \in \{0, 1, \ldots, |P| - 2\}$, we define $\text{glue}(P_iP_{i+1}) = (g, i)$, where g is the shared glue type between consecutive tiles P_i and P_{i+1} on the path P. Similarly, when we say "glue" in the context of a path P, we mean a pair of the form (glue type, path index). We define $\text{type}(\text{glue}(P_iP_{i+1})) = g$ to denote the glue type of $\text{glue}(P_iP_{i+1})$. We say that $\text{glue}(P_iP_{i+1})$ is *pointing to the north* (or *points to the north*, for short) if $\overrightarrow{P_iP_{i+1}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, *pointing to the west* if $\overrightarrow{P_iP_{i+1}} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$, *pointing to the south* if $\overrightarrow{P_iP_{i+1}} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$, and *pointing to the east* if $\overrightarrow{P_iP_{i+1}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

2.3 Fragile Paths

If two paths, or two assemblies, or a path and an assembly, share a common position we say that they *intersect* at that position. Furthermore, we say that two paths, or two assemblies, or a path and an assembly, *agree* on a position if they both place the same tile type at that position and *conflict* if they place a different tile type at that position. We say that a path P is *fragile* to mean that there is a producible assembly α that conflicts with P (intuitively, if we grow α first, then there is at least one tile that P cannot place), or more formally:

Definition 2.5 (Fragile). Let $\mathcal{T} = (T, \sigma, 1)$ be a tile assembly system and $P \in \text{P}[\mathcal{T}]$. We say that P is fragile if there exists a producible assembly $\alpha \in \mathcal{A}[\mathcal{T}]$ and a position $(x, y) \in (\text{dom}(\alpha) \cap \text{dom}(\text{asm}(P)))$ such that $\alpha((x, y)) \neq \text{asm}(P)((x, y))$.[6]

2.4 Pumping a Path

Next, for a path P and two indices i, j on P, we define a sequence of points and tile types (not necessarily a path) called the *pumping of P between i and j*:

[3] I.e. $\text{asm}(P)$ is a partial function from $\mathbb{Z}^2$ to tile types, and is defined on a connected set.

[4] Formally, non-intersection of a path $P = P_0P_1, \ldots$ and a seed assembly σ is defined as: $\forall t$ such that $t \in \sigma$, $\nexists i$ such that $\text{pos}(P_i) = \text{pos}(t)$.

[5] Intuitively, although producible paths are not assemblies, any producible path P has the nice property that it encodes an unambiguous description of how to grow $\text{asm}(P)$ from the seed σ, in path (P) order, to produce the assembly $\text{asm}(P) \cup \sigma$.

[6] Here, it might be the case that α and P conflict at only one position by placing two different tile types t and t', but that t and t' may place the same glues along P. In this case P is not producible when starting from the assembly α because one of the tiles along the positions of P is of the wrong type.

Definition 2.6 (Pumping of P between i and j). Let $\mathcal{T} = (T, \sigma, 1)$ be a tile assembly system and $P \in \mathrm{P}[\mathcal{T}]$. We say that the "*pumping of P between i and j*" is the sequence $\overline{q}$ of elements from $\mathbb{Z}^2 \times T$ defined by:

$$\overline{q}_k = \begin{cases} P_k & \text{for } 0 \leq k \leq i \\ P_{i+1+((k-i-1) \mod (j-i))} + \left\lfloor \frac{k-i-1}{j-i} \right\rfloor \overrightarrow{P_i P_j} & \text{for } i < k \end{cases}$$

Intuitively, $\overline{q}$ is the concatenation of a finite path $P_{0,1,\ldots,i}$ and an infinite periodic sequence of tile types and positions (possibly intersecting $\sigma \cup P_{0,1,\ldots,i}$, and possibly intersecting itself). We formalise this intuition in Lemma B.2 of the full version of this paper [45].

The following definition gives the notion of pumpable path used in our proofs. It is followed by a less formal but more intuitive description.

Definition 2.7 (Pumpable path). Let $\mathcal{T} = (T, \sigma, 1)$ be a tile assembly system. We say that a producible path $P \in \mathrm{P}[\mathcal{T}]$, is *infinitely pumpable*, or simply *pumpable*, if there are two integers $i < j$ such that the pumping of P between i and j is an infinite producible path, i.e. formally: $\overline{q} \in \mathrm{P}[\mathcal{T}]$.

In this case, we say that the *pumping vector* of $\overline{q}$ is $\overrightarrow{P_i P_j}$, and that P is *pumpable with pumping vector* $\overrightarrow{P_i P_j}$.

For a path P to be pumpable between i and j implies that $P_{i+1} + \overrightarrow{P_i P_j}$ interacts with P_j. It also implies that $\overline{q}$ is self-avoiding and that in particular, for any positive integers $s \neq t$, the path $P_{i+1,\ldots,j} + s\overrightarrow{P_i P_j}$ does not intersect with the path $P_{i+1,\ldots,j} + t\overrightarrow{P_i P_j}$. Lemma B.1 of the full version of this paper [45] shows that a sufficient condition for this is that $P_{i+1,\ldots,j}$ does not intersect $P_{i+1,\ldots,j} + \overrightarrow{P_i P_j}$.

2.5 2D Plane

2.5.1 Column, Glue Column, Row, Glue Row. When referring to sets of positions, we use the term "*the column x*" for some fixed $x \in \mathbb{Z}$ to mean the set $\{(x, y) \mid y \in \mathbb{Z}\}$, and the term "*the row y*" for some fixed $y \in \mathbb{Z}$ to mean the set $\{(x, y) \mid x \in \mathbb{Z}\}$. The *glue column x*, for some fixed $x \in \mathbb{Z}$, is the set of 2D half-integer positions $\{(x + 0.5, y) \mid y \in \mathbb{Z}\}$. The *glue row y*, for some fixed $y \in \mathbb{Z}$, is the set of 2D half-integer positions $\{(x, y + 0.5) \mid x \in \mathbb{Z}\}$.

Using the canonical embedding of $\mathbb{Z}^2$, the definition of a *glue column x* can also be defined as the set of edges of the grid graph of $\mathbb{Z}^2$ between column x and column $x+1$, and the *glue row y* is the set of edges of the grid graph of $\mathbb{Z}^2$ between row x and row $x + 1$. Which definition we use will always be clear from the context.

2.5.2 Curves. A curve $c : I \to \mathbb{R}^2$ is a function from an interval $I \subset \mathbb{R}$ to $\mathbb{R}^2$, where I is one of a closed, open, or half-open. All the curves in this paper are polygonal, i.e. unions of line segments and rays.

For a finite path P, we call the *embedding* $\mathfrak{E}[P]$ *of* P the curve defined for all $t \in [0, |P| - 1] \subset \mathbb{R}$ by:

$$\mathfrak{E}[P](t) = \mathrm{pos}(P_{\lfloor t \rfloor}) + (t - \lfloor t \rfloor)\overrightarrow{P_{\lfloor t \rfloor} P_{\lfloor t \rfloor + 1}}$$

Similarly, for a finite binding path p, the *embedding* $\mathfrak{E}[p]$ *of* p is the curve defined for all $t \in [0, |p| - 1] \subset \mathbb{R}$ by:

$$\mathfrak{E}[p](t) = p_{\lfloor t \rfloor} + (t - \lfloor t \rfloor)\overrightarrow{p_{\lfloor t \rfloor} p_{\lfloor t \rfloor + 1}}$$

The ray of vector $\overrightarrow{v}$ *from* (or, *that starts at*) point $A \in \mathbb{R}$ is defined as the curve $r : [0, +\infty[\to \mathbb{R}^2$ such that $r(t) = A + t\overrightarrow{v}$. The *vertical ray from a point A to the south (respectively to the north)* is the ray of vector $(0, -1)$ (respectively $(0, 1)$) from A, and the *horizontal ray from a point A to the west (respectively to the east)* is the ray of vector $(-1, 0)$ (respectively $(1, 0)$) from A.

If C is a curve defined on some real interval of the form $[a, b]$ or $]a, b]$, and D is a curve defined on some real interval of the form $[c, d]$ or $[c, d[$, and moreover $C(b) = D(c)$, then the *concatenation* $\mathrm{concat}(C, D)$ *of* C *and* D is the curve defined on $\mathrm{dom}(C) \cup (\mathrm{dom}(D) - (c - b))$ by:[7]

$$\mathrm{concat}(C, D)(t) = \begin{cases} C(t) \text{ if } t \leq b \\ D(t + (c - b)) \text{ otherwise} \end{cases}$$

A curve c is said to be *simple* or *self-avoiding* if all its points are distinct, i.e. if for all $x, y \in \mathrm{dom}(c)$, $c(x) = c(y) \Rightarrow x = y$.

For $a, b \in \mathbb{R}$ with $a \leq b$, the notation $[a, b]$ denotes a closed real interval, $]a, b[$ an open real interval, and $[a, b[$ and $]a, b]$ are open on one end and closed on the other. The reverse $c^{\leftarrow}$ of a curve c defined on some interval $[a, b]$ (respectively $[a, b[$, $]a, b]$, $]a, b[$) is the curve defined on $[-b, -a]$ (respectively $]-b, -a]$, $[-b, -a[$, $]-b, -a[$) as $c^{\leftarrow}(t) = c(-t)$.

If $A = (x_a, y_a) \in \mathbb{R}^2$ and $B = (x_b, y_b) \in \mathbb{R}^2$, the *segment* $\left[A, B\right]$ is defined to be the curve $s : [0, 1] \to \mathbb{R}^2$ such that for all $t \in [0, 1]$, $s(t) = ((1-t)x_a + tx_b, (1-t)y_a + ty_b)$. We sometimes abuse notation and write $\left[A, B\right]$ even if A or B (or both) is a *tile*, in which case we mean the position of that tile instead of the tile itself.

For a curve $c : \mathbb{R} \to \mathbb{R}^2$ we write $c(\mathbb{R})$ to denote the range of c (whenever we use this notation the curve c has all of $\mathbb{R}$ as its domain). When it is clear from the context, we sometimes write c to mean $c(\mathbb{R})$, for example

For a path or binding path, $p_{0,1,\ldots,k}$ of length ≥ 1, for $0 \leq i < k$ the notation $\mathrm{mid}(p_i p_{i+1})$ denotes the midpoint of the unit-length line segment $\mathfrak{E}[p_{i,i+1}]$. For a path P, we have $\mathrm{mid}(p_i p_{i+1}) = \mathrm{pos}(\mathrm{glue}(P_i P_{i+1}))$, hence this notation is especially useful for binding paths, since they do not have glues.

2.5.3 Cutting the Plane with Curves; Left and Right Turns. In this paper we use finite and infinite polygonal curves to cut the $\mathbb{R}^2$ plane into two pieces. The finite polygonal curves we use consist of a finite number of concatenations of vertical and horizontal segments of length 1 or 0.5. If the curve is simple and closed we may apply the Jordan Curve Theorem to cut the plane into connected components.

Theorem 2.8 (Jordan Curve Theorem). *Let c be a simple closed curve, then c cuts* $\mathbb{R}^2$ *into two connected components.*

Here, we have stated the theorem in its general form, although for our results the (easier to prove) polygonal version suffices.

The second kind of curve we use is composed of an infinite ray, then a finite number of length 0.5 or length 1 segments, and then another infinite ray. For such infinite polygonal curves we also

[7] $\mathrm{dom}(D) - (c - b)$ means $[b, d - (c - b)]$ if $\mathrm{dom}(D) = [c, d]$, and $[b, d - (c - b)[$ if $\mathrm{dom}(D) = [c, d[$

state and prove a slightly different version of the polygonal Jordan Curve Theorem, as Theorem B.3 in the full version of this paper [45].

In Appendix B of the full version of this paper [45] we define what it means for one curve to turn left or right from another, as well as left hand side and right hand side of a cut of the real plane.

2.5.4 Visibility. Let P be a path producible by some tile assembly system $\mathcal{T} = (T, \sigma, 1)$, and let $i \in \{0, 1, \ldots, |P| - 2\}$ be such that $\text{glue}(P_iP_{i+1})$ points east or west. We say that $\text{glue}(P_iP_{i+1})$ is *visible from the south* if and only if the ray ℓ^i of vector $(0, -1)$ starting at $\ell^i(0) = \left(\frac{x_{P_i}+x_{P_{i+1}}}{2}, y_{P_i}\right)$ does not intersect $\mathfrak{E}[P]$ nor σ.[8]

We define the terms *visible from the east*, *visible from the west* and *visible from the north* similarly.

In many of our proofs, we will use curves, in particular curves that include visibility ray(s), to define connected components. For example, consider the curve e, where we have a path $P_{i+1,i+2,\ldots,j,j+1}$ with $i < j$ and two glues $\text{glue}(P_iP_{i+1})$ and $\text{glue}(P_jP_{j+1})$ which are visible from the south with respective visibility rays l^i and l^j:

$$e = \text{concat}\left(l^{i\leftarrow}, \left[l^i(0), \text{pos}(P_{i+1})\right], \mathfrak{E}[P_{i+1,\ldots,j}], \left[\text{pos}(P_j), l^j(0)\right], l^j\right)$$

The curve e is defined on $]-\infty, +\infty[= \mathbb{R}$.

We will use the following lemma about visibility, which was stated in [31] (it is the fusion of Lemmas 5.2 and 6.3 in that paper). For the sake of completeness, we prove this result again, with slightly different notation.

Lemma 2.9. *Let P be a path producible by some tile assembly system $\mathcal{T} = (T, \sigma, 1)$ such that the last glue of P is visible from the north. Let $i, j \in \{0, 1, \ldots, |P| - 2\}$ be two integers. If both $\text{glue}(P_iP_{i+1})$ and $\text{glue}(P_jP_{j+1})$ are visible from the south and $\text{glue}(P_iP_{i+1})$ points to the east (respectively to the west), and $x_{P_i} < x_{P_j}$ (respectively $x_{P_i} > x_{P_j}$), then $i < j$ and $\text{glue}(P_jP_{j+1})$ points to the east (respectively to the west).*

We will sometimes use this lemma when the last tile of P is the unique easternmost tile of $\sigma \cup \text{asm}(P)$, in this case the last glue of P is visible from the south and from the north.

3 SHIELD LEMMA

In this section we state Lemma 3.2, our main technical tool. The following definition, illustrated in Figure 1, is crucial to the lemma statement:

Definition 3.1 (A shield (i, j, k) for P). Let P be a path producible by some tile assembly system $\mathcal{T} = (T, \sigma, 1)$. We say that the triple (i, j, k) of integers is a *shield for P* if $0 \le i < j \le k < |P| - 1$, and the following three conditions hold:

(1) $\text{glue}(P_iP_{i+1})$ and $\text{glue}(P_jP_{j+1})$ are both of the same type, visible from the south relative to P and pointing east; and
(2) $\text{glue}(P_kP_{k+1})$ is visible from the north relative to P, for notation let l^k be the visibility ray to the north of $\text{glue}(P_kP_{k+1})$; and
(3) $\mathfrak{E}[P_{i,i+1,\ldots,k}] \cap (l^k + \overrightarrow{P_jP_i}) \subseteq \{l^k(0) + \overrightarrow{P_jP_i}\}$. In other words, if $\mathfrak{E}[P_{i,i+1,\ldots,k}]$ intersects $l^k + \overrightarrow{P_jP_i}$ (which may not be the case), there is exactly one intersection, which is at the start-point of the ray $l^k + \overrightarrow{P_jP_i}$.

Lemma 3.2 (Shield lemma). *Let P be a path producible by some tile assembly system $\mathcal{T} = (T, \sigma, 1)$, such that (i, j, k) is a shield for P (see Definition 3.1). Then P is pumpable with pumping vector $\overrightarrow{P_iP_j}$, or P is fragile.*

Moreover, if P is fragile, there is a path Q, entirely contained in the workspace of shield (i, j, k), such that $P_{0,1,\ldots,i}Q$ is a producible path and conflicts with $P_{i+1,i+2,\ldots,k}$.

Intuition for the proof of Lemma 3.2. Starting from a producible path P with the properties described in Definition 3.1, we define three indices on P (called a "shield") which in turn are used to define an infinite curve c that partitions $\mathbb{R}^2$. Then we build a path R that stays on the right hand side of c, and that will ultimately allow us to reason about P and show that P is pumpable or fragile. Since the proof is rather involved, we split it up into parts, each containing one or more Claims:

- Using Definition 3.1, we define a bi-infinite curve c using the ray l^i, the path $P_{i+1,i+2,\ldots,k}$ and the ray l^k (see Figure 2). In the full version of this paper [45], we prove that c cuts $\mathbb{R}^2$ into two pieces: the left-hand and right-hand side of c. In the rest of the proof, we use the right-hand side $C \subset \mathbb{R}^2$ of c as a "workspace" where we can edit paths freely. The intuition is that c "shields our edits" from $\sigma \cup \text{asm}(P_{0,1,\ldots,i})$, which is entirely in the left-hand side of c, and thus prevents $\sigma \cup \text{asm}(P_{0,1,\ldots,i})$ from blocking these paths in the workspace C.
- We then reason by induction on the length of P. The initial setup for the inductive argument goes as follows (in a number of places we may reach the early conclusion that P is fragile, in which case we are done with the entire proof of Lemma 3.2):
  - We define a tile P_{m_0} of P called a *dominant tile*, see Figure 3. We then show that P_{m_0} is such that for all integers $n \ge 0$, $P_{m_0} + n\overrightarrow{P_iP_j}$ is in C.
  - Then, we define a binding path r in $\mathbb{Z}^2$ and prove a number of key properties about about it and its translation $r + \overrightarrow{P_jP_i}$ (see Figure 4). We then use r as a sequence of locations along which we we can either tile a producible path R, or else show that P is fragile. The path R is built in such a way that both R and $R + \overrightarrow{P_jP_i}$ are producible and in C.
  - To complete the setup for the inductive argument, we use R and m_0 to define the initial inductive indices u_0 and v_0 (see Figure 5 for an example). To define these indices, we use a ray L^{m_0} that starts from the tile P_{m_0} and splits the component C into two parts (called C^+ and C^-), which guarantees that $u_0 \le m_0 \le v_0$, which in turn means that the pumping of P between u_0 and v_0 is well-defined (i.e. $u_0 < v_0$) and has pumping vector $\overrightarrow{P_iP_j}$. (note that the pumping is not a simple path until the last step of the

[8] For a glue ray (whose range has half-integer x-coordinate), to not intersect σ (whose tiles are on integer points), we mean that the glue ray does not intersect any segment between two adjacent tiles of σ (even if these adjacent tiles do not interact).

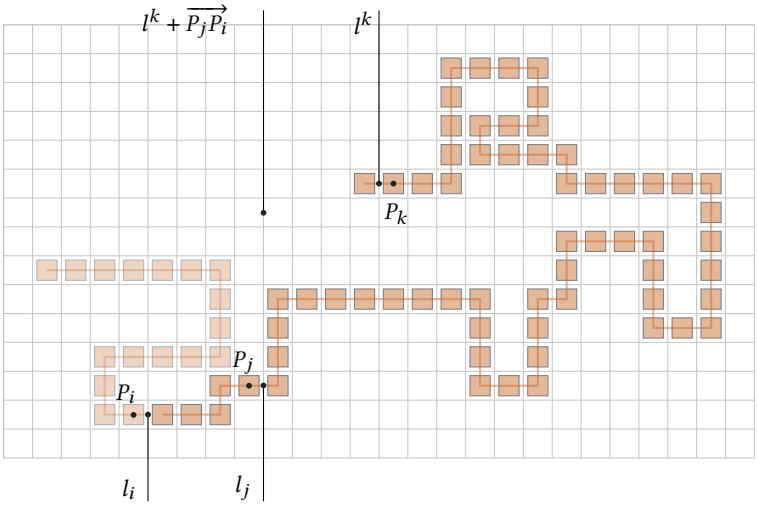


Figure 1: A suffix $P_{i,i+1,\ldots,k+1}$ of a path P. Tiles P_i, P_j and P_k are shown along with the four rays and the three glues (at ray starting points) of Hypotheses 1–3 of Definition 3.1, thus (i, j, k) is a shield for P.

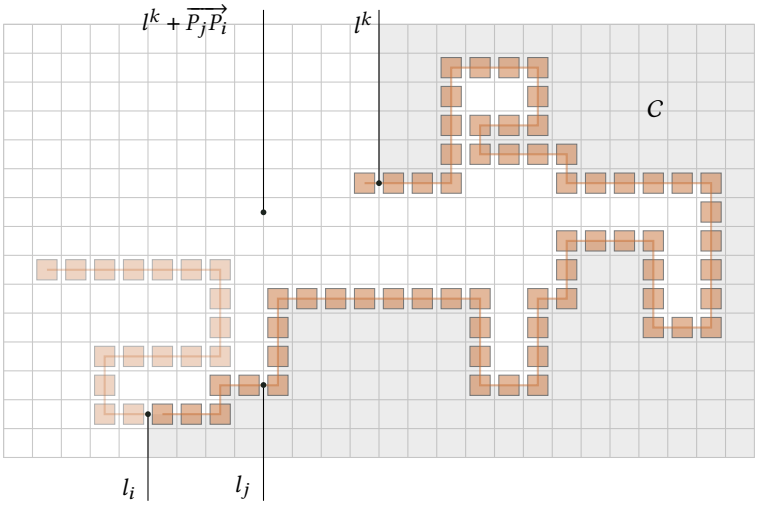


Figure 2: The path and shield triple (i, j, k) from Figure 1, annotated with curve c and component C. The border of the shaded region is the curve c, and the shaded region itself is the component C.

induction, where we eventually find a simple pumping of P).

In the inductive step, we show that either P is pumpable or fragile, or else we can use R again, along with inductive indices u_n, m_n, v_n, to find new indices u_{n+1}, m_{n+1} and v_{n+1}, but with $m_{n+1} > m_n$. Since P is of finite length, we will eventually run out of new indices (values for m_{n+1}, in particular), leading to the conclusion that P is either pumpable or fragile.

4 INTUITION AND ROADMAP FOR THE PROOF OF THEOREM 1.1

For the proof [45] of Theorem 1.1, we need to find three indices $i, j, k \in \{0, 1, \ldots, |P| - 1\}$ of P that satisfy the hypotheses of the Shield Lemma (Lemma 3.2, Definition 3.1), the conclusion of which is that P is pumpable or fragile. Throughout the proof [45] we apply the Shield Lemma in several different ways. We proceed in three steps:

(1) First, we make some trivial modifications to P (a rotation and translation of our frame of reference, and a truncation) so that P is in a canonical form where it reaches far to the east (P's final tile is to the east of the seed σ). We then invoke

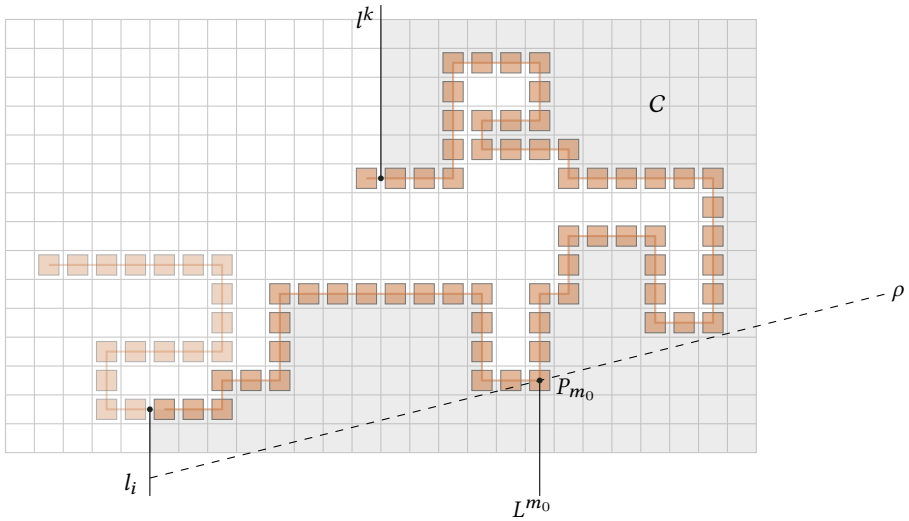


Figure 3: The ray ρ and tile P_{m_0}. We define ρ as the southernmost ray of vector $\overrightarrow{P_iP_j}$ that starts on l^i and intersects the position of at least one tile of $P_{i+1,i+2,\ldots,k}$. The easternmost such intersection is then defined to be $\mathrm{pos}(P_{m_0})$, and P_{m_0} is called the dominant tile.

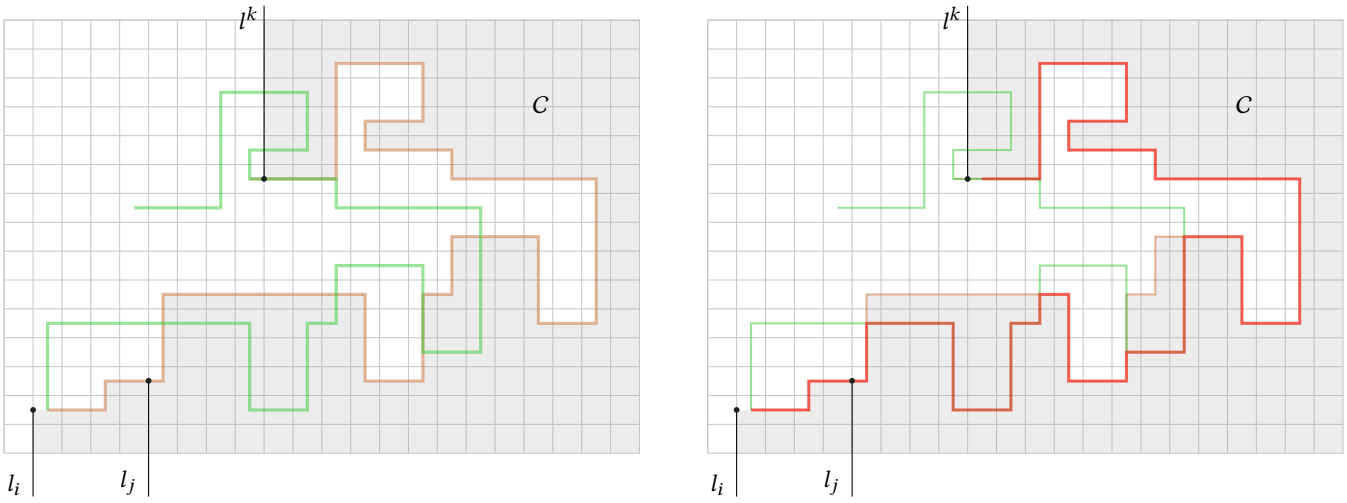


Figure 4: Left: Following from Figure 2, $\mathfrak{E}[P_{i+1,i+2,\ldots,k+1}]$ is shown in brown and $\mathfrak{E}[P_{j+1,j+2,\ldots,k+1}] + \overrightarrow{P_jP_i}$ is shown in green. Right: The route traced by (the embedding of) the binding path r: red indicates when r takes positions from $P_{i+1,i+2,\ldots,k}$ only, and brown indicates when r takes positions from $P_{j+1,j+2,\ldots,k} + \overrightarrow{P_jP_i}$ and/or $P_{i+1,i+2,\ldots,k}$. It turns out that either the binding path r can be tiled to give a producible path R, or else P conflicts with R, or conflicts with R's forward translation $R + \overrightarrow{P_iP_j}$, in such a way that means P is fragile and we are done with the proof of Lemma 3.2.

Theorem 1.1, the combinatorial-based proof of which goes through the following steps.

(2) Then, we show that either P is pumpable or fragile (in turn, by applying Lemma 3.2), or else that at most $|T|+1$ glues of P that are visible from the south or from the north can be pointing west (the remaining $\geq (4|T|)^{(4|T|+1)}(4|\sigma|+6) - |T| - |\sigma|$ glues visible from the south are pointing east).

(3) Then, we introduce the notion of *spans* (see Figure 6), and show that if we find two spans $S = (s, n)$ and $S' = (s', n')$ of the same orientation and type, and such that $s < s'$ and the height of S' is at least the height of S, then we can apply Lemma 3.2 to P, proving that P is pumpable or fragile.

(4) Finally, the proof of Theorem 1.1 uses a combinatorial argument, showing that if the path is long enough, there are enough spans that we can always find two spans of the same

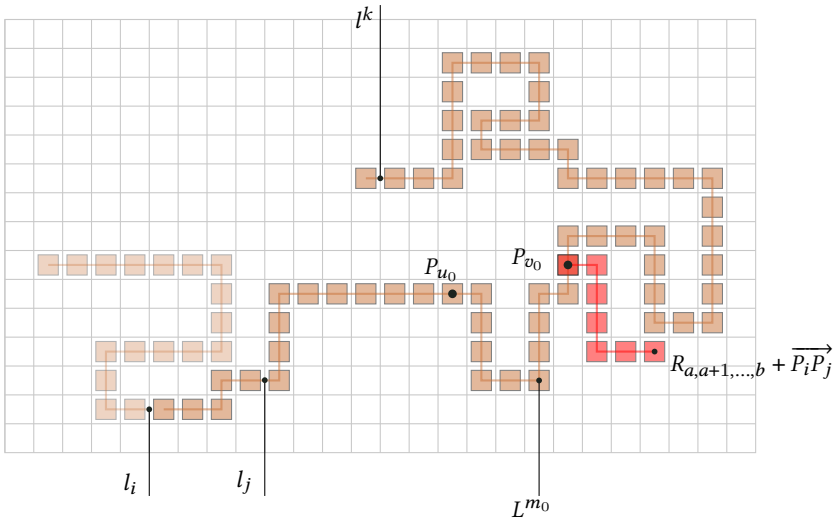


Figure 5: Growing the path $R + \overrightarrow{P_iP_j}$ and defining the base case (indices u_0, m_0 and v_0) for an inductive argument in the proof of Lemma 3.2. Following from Figure 4, the path R can be grown forward-translated by $\overrightarrow{P_iP_j}$ (a few ties of which are shown in red), and this in turn is used to define the indices u_0 and v_0. The path segment $R_{a,a+1,\ldots,b} + \overrightarrow{P_iP_j}$ shares its start position $\mathrm{pos}(R_a + \overrightarrow{P_iP_j})$ with the tile P_{v_0} and $\mathrm{pos}(P_{u_0}) = \mathrm{pos}(R_a)$. In the proof of Lemma 3.2, we find that $R_{a,a+1,\ldots,b} = P_{u_0,u_0+1,\ldots,m_0}$, and we go on to show that $u_0 \leq m_0 \leq v_0$, $P_{u_0} = P_{v_0} + \overrightarrow{P_jP_i}$, which establishes key facts for the base case of the inductive argument. In that inductive argument, intuitively, for any index i on P, and given a valid triple of indices (u_i, m_i, v_i) we either find that either P is pumpable or fragile, or else that there is a subsequent triple $(u_{i+1}, m_{i+1}, v_{i+1})$, found via growth of a suitable path in C. But since P is of finite length we can not encounter the latter case forever, thus P is pumpable or fragile. See [45] for details.

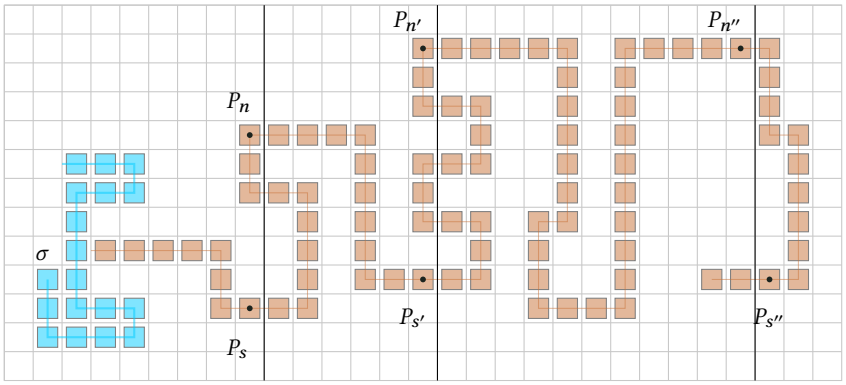


Figure 6: The seed is in blue, and P is in brown. Three example spans are shown in this figure, $S = (s, n)$, $S' = (s', n')$ and $S'' = (s'', n'')$. A span (s, n) is a pair of indices such that $\mathrm{glue}(P_sP_{s+1})$ is visible from the south, $\mathrm{glue}(P_nP_{n+1})$ is visible from the north, and $\mathrm{glue}(P_sP_{s+1})$ and $\mathrm{glue}(P_nP_{n+1})$ are on the same glue column. Here, both S and S' have both of their glues pointing east, hence we say that the span is pointing east. Moreover, since $s < n$ and $s' < n'$, S and S' are "up spans". On the other hand, S'' has its south glue pointing east, and its north glue pointing west, hence S'' is not pointing in any particular direction. Moreover, $n'' < s''$, hence S'' is a "down span". Span S has height 6, and spans S' and S'' have height 8. If the spans (s, n) and (s', n') happen to be of the same type (meaning $\mathrm{type}(\mathrm{glue}(P_sP_{s+1})) = \mathrm{type}(\mathrm{glue}(P_{s'}P_{s'+1}))$), since the height of S' is at least the height of S, we could apply Lemma 3.2 directly to prove that P is pumpable or fragile.

type and orientation, and of increasing height. This allows us to use Lemma 3.2 to conclude that P is pumpable or fragile.

REFERENCES

[1] Leonard Adleman, Qi Cheng, Ashish Goel, and Ming-Deh Huang. 2001. Running time and program size for self-assembled squares. In *STOC: Proceedings of the*

33rd Annual ACM Symposium on Theory of Computing. Hersonissos, Greece, 740–748. https://doi.org/10.1145/380752.380881

[2] Robert D Barish, Paul WK Rothemund, and Erik Winfree. 2005. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano letters* 5, 12 (2005), 2586–2592.

[3] Robert D Barish, Rebecca Schulman, Paul WK Rothemund, and Erik Winfree. 2009. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences* 106, 15 (2009), 6054–6059.

[4] B. Behsaz, J. Maňuch, and L. Stacho. 2012. Turing universality of step-wise and stage assembly at Temperature 1. In *DNA18: Proc. of International Meeting on DNA Computing and Molecular Programming (LNCS)*, Vol. 7433. Springer, 1–11.

[5] Hugues Bersini and Vincent Detours. 1994. Asynchrony induces stability in cellular automata based models. In *Artificial life IV*. MIT Press, MA, 382–387.

[6] Sarah Cannon, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Matthew J. Patitz, Robert Schweller, Scott M. Summers, and Andrew Winslow. 2013. Two Hands Are Better Than One (up to constant factors). In *STACS: Proceedings of the Thirtieth International Symposium on Theoretical Aspects of Computer Science*. LIPIcs, 172–184. Arxiv preprint: 1201.1650.

[7] Cameron T Chalk, Bin Fu, Alejandro Huerta, Mario A Maldonado, Eric Martinez, Robert T Schweller, and Tim Wylie. 2015. Flipping Tiles: Concentration Independent Coin Flips in Tile Self-Assembly. In *DNA21: Proceedings of the 21st International Conference on DNA Computing and Molecular Programming (LNCS)*, Andrew Phillips and Peng Yin (Eds.), Vol. 9211. Springer, 87–103.

[8] Qi Cheng, Gagan Aggarwal, Michael H. Goldwasser, Ming-Yang Kao, Robert T. Schweller, and Pablo Moisset de Espanés. 2005. Complexities for Generalized Models of Self-Assembly. *SIAM J. Comput.* 34 (2005), 1493–1515.

[9] Matthew Cook. 2004. Universality in elementary cellular automata. *Complex systems* 15, 1 (2004), 1–40.

[10] Matthew Cook, Yunhui Fu, and Robert T. Schweller. 2011. Temperature 1 self-assembly: deterministic assembly in 3D and probabilistic assembly in 2D. In *SODA: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*. 570–589. Arxiv preprint: arXiv:0912.0027.

[11] Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Matthew J. Patitz, Robert T. Schweller, Andrew Winslow, and Damien Woods. 2014. One Tile to Rule Them All: Simulating Any Tile Assembly System with a Single Universal Tile. In *ICALP: Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (LNCS)*, Vol. 8572. Springer, 368–379. Arxiv preprint: arXiv:1212.4756.

[12] Erik D. Demaine, Matthew J. Patitz, Trent A. Rogers, Robert T. Schweller, Scott M. Summers, and Damien Woods. 2013. The two-handed tile assembly model is not intrinsically universal. In *ICALP: Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (LNCS)*, Vol. 7965. Springer, 400–412. Arxiv preprint: arXiv:1306.6710.

[13] David Doty. 2012. Theory of algorithmic self-assembly. *Commun. ACM* 55, 12 (Dec. 2012), 78–88. https://doi.org/10.1145/2380656.2380675

[14] David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods. 2012. The tile assembly model is intrinsically universal. In *FOCS: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science* (New Brunswick, New Jersey). IEEE, 439–446. Arxiv preprint: arXiv:1111.3097.

[15] David Doty, Matthew J. Patitz, Dustin Reishus, Robert T. Schweller, and Scott M. Summers. 2010. Strong Fault-Tolerance for Self-Assembly with Fuzzy Temperature. In *FOCS: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*. 417–426.

[16] David Doty, Matthew J. Patitz, and Scott M. Summers. 2011. Limitations of self-assembly at temperature 1. *Theoretical Computer Science* 412, 1–2 (2011), 145–158. Arxiv preprint: arXiv:0903.1857v1.

[17] Constantine Glen Evans. 2014. *Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*. Ph.D. Dissertation. California Institute of Technology.

[18] Sándor P. Fekete, Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Robert T. Schweller. 2015. Universal Computation with Arbitrary Polyomino Tiles in Non-Cooperative Self-Assembly. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 148–167. http://arxiv.org/abs/1408.3351

[19] Bin Fu, Matthew J. Patitz, Robert T. Schweller, and Robert Sheline. 2012. Self-assembly with Geometric Tiles. In *ICALP: Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (LNCS)*, Vol. 7391. Springer, 714–725.

[20] Kenichi Fujibayashi, Rizal Hariadi, Sung Ha Park, Erik Winfree, and Satoshi Murata. 2007. Toward Reliable Algorithmic Self-Assembly of DNA Tiles: A Fixed-Width Cellular Automaton Pattern. *Nano Letters* 8, 7 (2007), 1791–1797.

[21] David Furcy and Scott M Summers. 2018. Optimal Self-Assembly of Finite Shapes at Temperature 1 in 3D. *Algorithmica* 80, 6 (2018), 1909–1963.

[22] David Furcy, Scott M Summers, and Christian Wendlandt. 2019. New Bounds on the Tile Complexity of Thin Rectangles at Temperature-1. In *DNA25: International Conference on DNA Computing and Molecular Programming*. Springer, 100–119.

[23] Oscar Gilbert, Jacob Hendricks, Matthew J Patitz, and Trent A Rogers. 2016. Computing in continuous space with self-assembling polygonal tiles. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 937–956. Arxiv preprint: arXiv:1503.00327.

[24] Jacob Hendricks, Matthew J. Patitz, Trent A. Rogers, and Scott M. Summers. 2014. The Power of Duples (in Self-Assembly): It's Not So Hip To Be Square. In *COCOON: Proceedings of 20th International Computing and Combinatorics Conference*. 215–226. Arxiv preprint: arXiv:1402.4515.

[25] Natasa Jonoska and Daria Karpenko. 2014. Active Tile Self-assembly, Part 1: Universality at temperature 1. *Int. J. Found. Comput. Sci.* 25, 2 (2014), 141–164. https://doi.org/10.1142/S0129054114500087

[26] Ján Maňuch, Ladislav Stacho, and Christine Stoll. 2010. Two lower bounds for self-assemblies at Temperature 1. *Journal of Computational Biology* 17, 6 (2010), 841–852.

[27] Pierre-Étienne Meunier. 2015. Non-cooperative Algorithms in Self-assembly. In *UCNC: Unconventional Computation and Natural Computation (LNCS)*, Vol. 9252. Springer, 263–276.

[28] Pierre-Étienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods. 2014. Intrinsic universality in tile self-assembly requires cooperation. In *SODA: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*. 752–771. Arxiv preprint: arXiv:1304.1679.

[29] Pierre-Étienne Meunier and Damien Regnault. 2015. A pumping lemma for noncooperative self-assembly. (2015). Arxiv preprint: arXiv:1312.6668v4 [cs.CC].

[30] Pierre-Étienne Meunier and Damien Regnault. 2019. Non-cooperatively assembling large structures. In *DNA Computing and Molecular Programming - 25th International Conference, DNA 25, Seattle, WA, USA, August 5-9, 2019, Proceedings*.

[31] Pierre-Étienne Meunier and Damien Woods. 2017. The non-cooperative tile assembly model is not intrinsically universal or capable of bounded Turing machine simulation. In *STOC: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, Montreal, Canada, 328–341. Arxiv preprint with full proofs: arXiv:1702.00353v2 [cs.CC].

[32] Turlough Neary and Damien Woods. 2006. P-completeness of cellular automaton Rule 110. In *ICALP: The 33rd International Colloquium on Automata, Languages and Programming (LNCS)*, Vol. 4051. Springer, 132–143.

[33] Jennifer E. Padilla, Matthew J. Patitz, Robert T. Schweller, Nadrian C. Seeman, Scott M. Summers, and Xingsi Zhong. 2014. Asynchronous Signal Passing for Tile Self-Assembly: Fuel Efficient Computation and Efficient Assembly of Shapes. *International Journal of Foundations of Computer Science* 25, 4 (2014), 459–488. Arxiv preprint: arxiv:1202.5012.

[34] Matthew J. Patitz. 2014. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing* 13(2) (2014), 195–224. https://doi.org/10.1007/s11047-013-9379-4

[35] Matthew J. Patitz, Robert T. Schweller, and Scott M. Summers. 2011. Exact Shapes and Turing Universality at Temperature 1 with a Single Negative Glue. In *DNA 17: Proceedings of the Seventeenth International Conference on DNA Computing and Molecular Programming (LNCS)*. Springer, 175–189. Arxiv preprint: arXiv:1105.1215.

[36] Paul W.K. Rothemund, Nick Papadakis, and Erik Winfree. 2004. Algorithmic Self-Assembly of DNA Sierpinski Triangles. *PLoS Biology* 2, 12 (2004), 2041–2053.

[37] Paul W. K. Rothemund. 2001. *Theory and Experiments in Algorithmic Self-Assembly*. Ph.D. Dissertation. University of Southern California.

[38] Paul W. K. Rothemund and Erik Winfree. 2000. The Program-size Complexity of Self-Assembled Squares (extended abstract). In *STOC: Proceedings of the thirty-second annual ACM Symposium on Theory of Computing*. ACM, Portland, Oregon, 459–468. https://doi.org/10.1145/335305.335358

[39] Rebecca Schulman, Bernard Yurke, and Erik Winfree. 2012. Robust self-replication of combinatorial information via crystal growth and scission. *Proceedings of the National Academy of Sciences* 109, 17 (2012), 6405–6410.

[40] David Soloveichik and Erik Winfree. 2007. Complexity of Self-Assembled Shapes. *SIAM J. Comput.* 36, 6 (2007), 1544–1569.

[41] Hao Wang. 1961. Proving Theorems by Pattern Recognition – II. *The Bell System Technical Journal* XL, 1 (1961), 1–41.

[42] Erik Winfree. 1998. *Algorithmic Self-Assembly of DNA*. Ph.D. Dissertation. California Institute of Technology.

[43] Damien Woods. 2015. Intrinsic universality and the computational power of self-assembly. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 373, 2046 (2015). dx.doi.org/10.1098/rsta.2014.0214.

[44] Damien Woods, David Doty, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. 2019. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature* 567, 7748 (2019), 366–372.

[45] Pierre Étienne Meunier, Damien Regnault, and Damien Woods. 2020. The program-size complexity of self-assembled paths. Arxiv preprint with full proofs: arXiv:2002.04012 [cs.CC].